SOAP Address Book API for txttools v6.4

Software Version - 6.4
Document Version - 1.6

John Hunsley / Andrew Alexander
aalexander@txttools.co.uk
Date – 01/08/2014

Contents

Document Change Log

1.1 AddressbookResponse schema changed in line with correct format for Object/XML mapping. See 3.1.3.

1.2 Null flag function added and required Contact Details sub elements set to non required. Overwrite and Remove elements corrected to boolean values. See 3.1.2.

1.3 Updating Group Data section added, sub groups description corrected, and disassociated sub groups element and example added. Known issues section added.

1.4 Known Issues section updated.

1.5 Overwrite element updated

1.6 Empty string value field blanking added, undocumented Contact Details sub elements added to the element overview table.

1 Introduction

1.1 Intended audience

This document is aimed at txttools customers who wish to manage their online address book remotely from their own application without using the txttools web user interface.

Before reading this document and implementing a software client the reader should have a solid understanding of the Hyper Text Transfer Protocol (HTTP) and Extensible Markup Language (XML). In particular, the reader should understand the HTTP request and request and response mechanisms and be able to create well formed XML strings of data.

For more information regarding these technologies please refer to the following guides:

HTTP
http://www.w3.org/Protocols/
http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

XML
http://www.xml.com/pub/a/98/10/guide0.html
http://en.wikipedia.org/wiki/XML

1.2 Useful tools

To leverage the usability of SOAP web services you will require a SOAP client capable of processing WSDL files and making SOAP calls to the API. Your aim is to integrate your application/database/MIS System with the txttools SOAP API, for which you will need to implement your own client. Java, .net, Oracle and many other platforms provide some sort of SOAP client support framework enabling you to create your own client as part of your application. You will however require a 'stand alone' SOAP client which you can use to test out the API and formulate the required XML to make a SOAP call. I suggest using a Java based client called the Soap UI. It is platform independent and requires the Java Runtime Environment installed which you should already have as part of a standard Windows, MAC or Linux operating system distribution. The Soap UI is free and may be downloaded from the following URI:

http://www.soapui.org/

There are many other SOAP clients out there, but for the purposes of the tutorials and examples in this document I will assume you are using the Soap UI.

You may also require some XML editing schema validation tools to help you create and validate your XML.

http://www.xmlcooktop.com/

If you would like to closely inspect the HTTP traffic between your client and the SOAP API then I suggest using the following application.

http://www.parosproxy.org

1.3 What is SOAP?

Simple **O**bject **A**ccess **P**rotocol is an implementation of Service Oriented Architecture. It is a standard method for exchanging messages in XML format between computers over a network, usually HTTP. XML data may be passed between computers over HTTP on the request and response. SOAP is an extension of this and defines a standard XML format and port of call for schema discovery. A SOAP API is usually accompanied by a Web Service Description Language (WSDL) file and it is this which should be your first port of call for information about the services available to the client.

For a detailed discussion of SOAP and WSDL files see:

http://en.wikipedia.org/wiki/SOAP
http://en.wikipedia.org/wiki/Web_Services_Description_Language

The WSDL file is an XML document which describes the ports and end points of services which may be invoked by a call and the XML schema expected by each service. The schema definitions show what data is required, the format it must be in and the nature of the response the client should expect when invoking an operation at an end point. This document is not meant to be human readable, however if you provide it to a client, such as the Soap UI, the file will be processed and a default request schema created for each service end point. The client requires only the WSDL file, the service end point name and the data in order to make a call to a service and invoke an operation.

1.4 Why use the SOAP API?

txttools provides basic XML over HTTP APIs for both messaging and address book functions. To use these APIs the client must provide an XML string in the expected schema on the HTTP request. The SOAP mechanism is exactly the same, its just some XML sent over HTTP on the request. So why bother using SOAP if we already have an XML/HTTP API?

The reason is simple! The SOAP client does not need to know anything about the expected XML schema prior to reading the WSDL, whereas an XML/HTTP API client must provide the XML in the schema outlined in the documentation for that API, the programmer creating the client must ensure the XML is hard coded with the known schema.

If you don't want to waste time hard coding String XML values into your client then use the SOAP API rather than the old basic XML/HTTP API.

2 Authentication

A client must authenticate before invoking any operation from the SOAP API. The client must provide the username and password in plain text for the txttools account they wish to use. If you are using the messaging API any messages you send will be sent from that account, any inbox messages you retrieve will come from that account and will be designated as 'read' once the API has successfully returned them in the SOAP response.

If you are using the address book API you will be managing contacts and groups from that account address book and any account which shares that address book. You must have permission to alter the address book before invoking any address book API operation.

All txttools SOAP APIs employ the standard Web Services Security (WSSE) specification to define authentication credentials. The specification is defined by Oasis Open Standards Consortium, you can find more information at the following URI:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

Clients must provide their credentials in plain text in the SOAP header of the message. It is strongly recommended you use HTTPS for all your communication with all txttools SOAP APIs, this will secure both your authentication credentials in the header and any information you provide in the SOAP body.

Clients must provide both their account username and password in the format outlined in the example below.

```
<soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <wsse:UsernameToken wsu:Id="UsernameToken-2750779"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
      utility-1.0.xsd">
      <wsse:Username>USERNAME</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">PASSWORD</wsse:Password>
    </wsse:UsernameToken>
  </wsse:Security>
</soapenv:Header>
```

3 The Address Book API

The txttools Address Book WSDL file can be found at:

http://www.txttools.co.uk/connectors/soap/addressbook/txttoolsAddressbook.wsdl

Processing the WSDL file reveals one service:

- txttoolsAddressbookServiceBinding

The service has one single operation:

- Addressbook

3.1 The address book operation

All address book management functions may be carried out by invoking this operation. Different functions are executed depending upon the XML elements you choose to include in the request body. The client may select to completely overwrite the existing address book data with the request data or update it.

The functions which may be executed by invoking this operation include:

- Add new/update/remove contact data to the address book
- Add new/update/remove groups/group data to the address book
- Add new or existing contacts to new or existing groups
- Disassociate existing contacts from a group
- Add new or existing sub groups to a group
- Add new or existing parent groups to a group
- Disassociate sub groups from a group
- Associate parents with a student contact – (Edutxt users only)
- Disassociate existing parents from a student contact – (Edutxt users only)

The following table describes the XML SOAP body elements which may be used within this operation. See the examples section (3.1.4) for details concerning which XML elements to include for different address book functions.

The namespace prefix for all elements for this operation is – **`sch`**

Element	Description & format	Required	Cardinality
`AddressbookRequest`	Contains all address book request XML sub elements.	Yes	1
`Overwrite`	Denotes whether the address book request should overwrite the entire current address book, overwrite the contacts only, or update without removing any	No	0 - 1

	previous data. True or all to overwrite all, contacts to only overwrite the contacts, false to update. Defaults to false if not supplied.		
Addressbook	Contains all address book data request XML sub elements.	Yes	1
Contact	Holds the information relating to a single new or existing address book contact within the **Addressbook** element.	No	0 - many per **Addressbook**
Remove	Denotes whether to remove the contact from the address book. True to remove or false to update. Defaults to false if omitted.	No	0 – 1 per **Contact**
ContactDetails	Contains all contact details XML sub elements such as name and phone number.	Yes, in the context of a **Contact**	1 per **Contact**
UniqueId	The unique identifier for this contact within the address book – alphanumeric characters only.	No	0 – 1 per **ContactDetails**
Firstname	The first name of the contact – alphanumeric characters only.	No	0 - 1 per **ContactDetails**
Lastname	The last name /second name /surname of the contact – alphanumeric characters only.	No	0 - 1 per **ContactDetails**
Phonenumber	The mobile phone number of the contact. 11-14 characters in length, either in international format starting with '+' or '00' followed by the international prefix, or in standard UK format starting with '0'. All following characters must	No	0 - 1 per **ContactDetails**

	be numeric.		
Email	The contact's email address. Must be a valid email address.	No	0 – 1 per **ContactDetails**
Notes	Any additional information you wish to supply for this contact. Max 255 alphanumeric characters.	No	0 – 1 per **ContactDetails**
Gender	The contact's gender – 'Male', 'Female', #NULL# tag or empty string	No	0 - 1 per **ContactDetails**
HouseNumber	The contact's house number. Max 255 alphanumeric characters.	No	0 - 1 per **ContactDetails**
Address1	The contact's address details. Max 255 alphanumeric characters.	No	0 - 1 per **ContactDetails**
Address2	The contact's additional address details. Max 255 alphanumeric characters.	No	0 - 1 per **ContactDetails**
City	The contact's city. Max 255 alphanumeric characters.	No	0 - 1 per **ContactDetails**
County	The contact's county. Max 255 alphanumeric characters.	No	0 - 1 per **ContactDetails**
Country	The contact's country. Max 255 alphanumeric characters.	No	0 - 1 per **ContactDetails**
Postcode	The contact's postcode. Max 255 alphanumeric characters.	No	0 - 1 per **ContactDetails**
DOB	The contact's date of birth., must be valid date in YYYY-MM-DD format, #NULL# tag or empty string.	No	0 - 1 per **ContactDetails**
AssociatedGroups	The groups for which this contact will be a member. If the group does not already exist it will be created. See **GroupDetails** element.	No	0 – 1 per **Contact**
DisassociatedGroups	The existing groups	No	0 – 1 per **Contact**

	within the address book for which this contact will be disassociated. See **GroupDetails** element for more details.		
AssociatedParents	The parent contacts for which this contact will be associated as a child. Applies to Edutxt users only. See **ContactDetails** element for more details.	No	0 – 1 per **Contact**
DisassociatedParents	The existing parent contacts for which this contact will be disassociated as a child. Applies to Edutxt users only. See **ContactDetails** element for more details.	No	0 – 1 per **Contact**
Group	Holds the information for a single new or existing group within the **Addressbook** element.	No	0 – many per **Addressbook**
Remove	Denotes whether to remove the group from the address book. True to remove or false to update. Defaults to false if omitted.	No	0 – 1per **Group**
GroupDetails	Holds the details of the group, including name and description.	Yes, in the context of **Group**	0 – 1 per **Group**
Name	The name of the group. Maximum 100 alphanumeric characters. The name is case insensitive if the client is referring to an existing object.	Yes, in the context of **GroupDetails**	1 per **GroupDetails**
Description	The description of the group. Maximum 255 alphanumeric characters.	No	0 – 1 per **GroupDetails**
Type	The named group type, either Group, Class or Year. Applies to Edutxt clients only. Defaults to	No	0 – 1 per **GroupDetails**

	Group if omitted.		
AssociatedContacts	The new or existing contacts to be members associated with this group. See **ContactDetails** for more information.	No	0 – 1 per **Group**
DisassociatedContacts	The existing contacts to be disassociated from this existing group. See **ContactDetails** for more information.	No	0 – 1 per **Group**
SubGroups	Sub groups contained within this group. The group hierarchy may be as deep as the client wishes. A group may contain as many sub groups as the client wishes and a child group may be a child of multiple parents. If a sub group is added to a group that does not contain other sub groups, the parent group loses any contacts it contains. See **GroupDetails** for more information about the sub elements to use here.	No	0 – 1 per **Group**
DisassociatedSubGroups	The existing sub groups to be disassociated from this existing parent group. See **GroupDetails** for more information.	No	0 – 1 per **Group**

3.1.1 Understanding Object Equality.

When updating or removing a contact from the address book it is essential to uniquely identify the contact object you wish to manage. The simplest way to do this is to ensure all contacts you upload to the address book are identified with a unique id. A unique id is not required by the operation and if you have not supplied a unique id, the contacts will try to be matched in the following order:

- Firstname, Lastname, Phone number
- Phone number
- Firstname, Lastname

Group objects are unique per address book by case insensitive name. If you wish to manage a

group, add or remove contacts, then you must identify that group by its name alone.

3.1.2 Updating contact data

When updating a contact within the address book you must identify the contact by supplying the contact's details in the **`ContactDetails`** request element as described in 3.1.1. If the contact in the address book does not match any of the details in 3.1.1, the contact **cannot** be updated via a SOAP request. Only specify fields you wish to update – omitted fields in the request will remain unchanged in the address book. In order to remove a field value from an existing contact within the address book i.e. set it null, you must specify that field with the null flag - **#NULL#**. Additionally self-closing tags **`<Firstname/>`** and empty value tags **`<Firstname></Firstname>`** can be used to remove a field value. Those fields which are validated by the SOAP API, such as email, gender and phone number will allow only those valid values or the null flag or an empty string value.

3.1.3 Updating group data

When updating a group within the address book you **must** identify the group by name. If the description field is not specified it will remain unchanged in the address book, alternatively a new description can be entered. In order to remove the description value from a group within the address book i.e. set it null, you must specify the field with the null flag - **#NULL#**. Additionally self-closing tags **`<Name/>`** and empty value tags **`<Name></Name>`** can be used to remove a field value.

3.1.4 Examples

Update the address book adding two new contacts:

```
<soapenv:Body>
  <sch:AddressbookRequest>
    <sch:Overwrite>false</sch:Overwrite>
    <sch:Addressbook>
      <sch:Contact>
        <sch:ContactDetails>
          <sch:UniqueId>1234567890</sch:UniqueId>
          <sch:Firstname>John</sch:Firstname>
          <sch:Lastname>Hunsley</sch:Lastname>
          <sch:Phonenumber>+447772211117</sch:Phonenumber>
          <sch:Email>jhunsley@txttools.co.uk</sch:Email>
          <sch:Notes>Technical Consultant</sch:Notes>
        </sch:ContactDetails>
      </sch:Contact>
      <sch:Contact>
        <sch:ContactDetails>
          <sch:UniqueId>0987654321</sch:UniqueId>
          <sch:Firstname>Olivier</sch:Firstname>
          <sch:Lastname>Tane</sch:Lastname>
          <sch:Phonenumber>+447817577380</sch:Phonenumber>
          <sch:Email>otane@txttools.co.uk</sch:Email>
          <sch:Notes>Support Manager</sch:Notes>
        </sch:ContactDetails>
      </sch:Contact>
    </sch:Addressbook>
  </sch:AddressbookRequest>
</soapenv:Body>
```

Update the address book and update an existing contact with a new phone number:

```
<soapenv:Body>
  <sch:AddressbookRequest>
    <sch:Overwrite>false</sch:Overwrite>
    <sch:Addressbook>
      <sch:Contact>
        <sch:ContactDetails>
          <sch:UniqueId>1234567890</sch:UniqueId>
          <sch:Phonenumber>+447884050705</sch:Phonenumber>
        </sch:ContactDetails>
      </sch:Contact>
    </sch:Addressbook>
  </sch:AddressbookRequest>
</soapenv:Body>
```

Update the address book and remove an existing contact:

```
<soapenv:Body>
  <sch:AddressbookRequest>
    <sch:Overwrite>false</sch:Overwrite>
    <sch:Addressbook>
      <sch:Contact>
        <sch:Remove>true</sch:Remove>
        <sch:ContactDetails>
          <sch:UniqueId>1234567890</sch:UniqueId>
        </sch:ContactDetails>
      </sch:Contact>
    </sch:Addressbook>
  </sch:AddressbookRequest>
</soapenv:Body>
```

Update the address book and add an existing contact to a new group whilst also removing that contact's email, phone number and notes values:

```
<soapenv:Body>
  <sch:AddressbookRequest>
    <sch:Overwrite>false</sch:Overwrite>
    <sch:Addressbook>
      <sch:Contact>
        <sch:ContactDetails>
          <sch:UniqueId>0987654321</sch:UniqueId>
          <sch:Email>#NULL#</sch:Email>
          <sch:Phonenumber/>
          <sch:Notes></sch:Notes>
        </sch:ContactDetails>
        <sch:AssociatedGroups>
          <sch:GroupDetails>
            <sch:Name>Support Team</sch:Name>
            <sch:Description>txttools support team</sch:Description>
          </sch:GroupDetails>
        </sch:AssociatedGroups>
      </sch:Contact>
    </sch:Addressbook>
  </sch:AddressbookRequest>
</soapenv:Body>
```

Update the address book and remove an existing contact from a group, whilst also changing the group's description:

```
<soapenv:Body>
  <sch:AddressbookRequest>
    <sch:Overwrite>false</sch:Overwrite>
    <sch:Addressbook>
      <sch:Contact>
        <sch:ContactDetails>
          <sch:UniqueId>0987654321</sch:UniqueId>
        </sch:ContactDetails>
        <sch:DisassociatedGroups>
          <sch:GroupDetails>
            <sch:Name>Support Team</sch:Name>
            <sch:Description>txttools support team group</sch:Description>
          </sch:GroupDetails>
        </sch:DisassociatedGroups>
      </sch:Contact>
    </sch:Addressbook>
  </sch:AddressbookRequest>
</soapenv:Body>
```

Remove an existing group from the address book:

```
<soapenv:Body>
  <sch:AddressbookRequest>
    <sch:Overwrite>false</sch:Overwrite>
    <sch:Addressbook>
      <sch:Group>
        <sch:Remove>true</sch:Remove>
        <sch:GroupDetails>
          <sch:Name>Support Team</sch:Name>
        </sch:GroupDetails>
      </sch:Group>
    </sch:Addressbook>
  </sch:AddressbookRequest>
</soapenv:Body>
```

Update the address book and add one new contact and one existing contact to two new groups as sub groups of another new parent group:

```
<soapenv:Body>
  <sch:AddressbookRequest>
    <sch:Overwrite>false</sch:Overwrite>
    <sch:Addressbook>
      <sch:Group>
        <sch:GroupDetails>
          <sch:Name>txttools Team</sch:Name>
          <sch:Description>the txttools team</sch:Description>
        </sch:GroupDetails>
        <sch:SubGroups>
          <sch:Group>
            <sch:GroupDetails>
              <sch:Name>Support Team</sch:Name>
              <sch:Description>the support team</sch:Description>
            </sch:GroupDetails>
            <sch:AssociatedContacts>
              <sch:ContactDetails>
                <sch:UniqueId>0987654321</sch:UniqueId>
              </sch:ContactDetails>
            </sch:AssociatedContacts>
          </sch:Group>
          <sch:Group>
            <sch:GroupDetails>
              <sch:Name>Technical Team</sch:Name>
              <sch:Description>the tech team</sch:Description>
            </sch:GroupDetails>
            <sch:AssociatedContacts>
              <sch:ContactDetails>
                <sch:UniqueId>1234567890</sch:UniqueId>
                <sch:Firstname>John</sch:Firstname>
                <sch:Lastname>Hunsley</sch:Lastname>
                <sch:Phonenumber>+447772211117</sch:Phonenumber>
                <sch:Email>jhunsley@txttools.co.uk</sch:Email>
                <sch:Notes>Technical Consultant</sch:Notes>
              </sch:ContactDetails>
            </sch:AssociatedContacts>
          </sch:Group>
        </sch:SubGroups>
      </sch:Group>
    </sch:Addressbook>
  </sch:AddressbookRequest>
</soapenv:Body>
```

Update the address book and add one new contact to a new group as a sub group of an existing group, and disassociate an existing sub group from the existing group:

```
<soapenv:Body>
  <sch:AddressbookRequest>
    <sch:Overwrite>false</sch:Overwrite>
    <sch:Addressbook>
      <sch:Group>
        <sch:GroupDetails>
          <sch:Name>txttools Team</sch:Name>
        </sch:GroupDetails>
        <sch:SubGroups>
          <sch:Group>
            <sch:GroupDetails>
              <sch:Name>Sales Team</sch:Name>
              <sch:Description>The sales team</sch:Description>
            </sch:GroupDetails>
            <sch:AssociatedContacts>
              <sch:ContactDetails>
                <sch:UniqueId>1112222333</sch:UniqueId>
                <sch:Firstname>Joe</sch:Firstname>
                <sch:Lastname>Bloggs</sch:Lastname>
                <sch:Phonenumber>+447812345678</sch:Phonenumber>
                <sch:Email>jbloggs@txttools.co.uk</sch:Email>
              </sch:ContactDetails>
            </sch:AssociatedContacts>
          </sch:Group>
        </sch:SubGroups>
        <sch:DisassociatedSubGroups>
          <sch:Group>
            <sch:GroupDetails>
              <sch:Name>Technical Team</sch:Name>
            </sch:GroupDetails>
          </sch:Group>
        </sch:DisassociatedSubGroups>
      </sch:Group>
    </sch:Addressbook>
  </sch:AddressbookRequest>
</soapenv:Body>
```

3.1.5 Response

If the client has successfully invoked the Addressbook operation and the function successfully carried out at txttools a success message is returned in the SOAP response.

There is only one element in a successful invocation response as detailed below. name space prefix for the body element is - **txttoolsAddressbook**

Element	Description & format	Required	Cardinality
AddressbookResponse	Wraps the address book response	Yes	1
Response	A string – 'Processed ok'	Yes	1

The invocation may not been successful for various reasons. If you have provided invalid authentication credentials then you will receive a standard WSSE security fault. You may not have permission to alter the address book, in which case you will receive a fault response stating the exception. If you receive anything other than the 'Processed ok' message you should assume the operation has not taken place.

3.1.6 Response example

A response following a successful invocation of the Addressbook operation:

```
<SOAP-ENV:Body>
  <txttoolsAddressbook:AddressbookResponse
xmlns:txttoolsAddressbook="http://www.txttools.co.uk/connectors/soap/addressbook
/schemas">
    <txttoolsAddressbook:Response>Processed ok</txttoolsAddressbook:Response>
  </txttoolsAddressbook:AddressbookResponse>
</SOAP-ENV:Body>
```

4 Known Issues

There is currently one known issue with the SOAP API. If you find any others, please contact us (details below).

- When deleting a group from the address book, any associated contacts do not get evicted from any of the parent groups until the contact or one of the groups is saved again.

5 Further assistance

If you require further assistance with the examples or any of the information covered in this document please contact the txttools team – 0113 234 2111 or email techies@txttools.co.uk.